Predição de Séries Temporais Baseada em Redes Neurais Artificiais

Aluno: João Pedro de Oliveira Pagnan [FEEC/UNICAMP]
Orientador: Prof. Levy Boccato [FEEC/UNICAMP]
Coorientador: Prof. Romis Ribeiro de Faissol Attux [FEEC/UNICAMP]

Palavras-chave – Redes Neurais Artificiais, Sistemas Caóticos, Predição, Séries Temporais

1. Introdução

A predição de séries temporais é uma das aplicações mais interessantes do tratamento de informação. O desafio de antecipar padrões de comportamento e construir modelos que sejam apropriados para explicar determinados fenômenos da natureza tem importância para a biologia, economia, automação industrial, meteorologia e diversas outras áreas da ciência [1].

Na literatura, encontramos diversos tipos de modelos para a predição de séries temporais, desde métodos clássicos lineares, como o modelo autorregressivo (AR) [1] até métodos não-lineares utilizando, por exemplo, redes neurais artificiais, sendo que dessas se destacam as redes do tipo *Multilayer Perceptron* (MLP) [2] e as redes recorrentes, especialmente a *Long Short-Term Memory* (LSTM) [3] e a *Echo State Network* (ESN) [4].

Uma classe de sistemas dinâmicos particularmente relevante dentro do contexto de modelagem e predição de séries temporais está ligada à ideia de dinâmica caótica. Diversos fenômenos naturais, como a dinâmica populacional de uma espécie, a dinâmica atmosférica de uma região, ou até mesmo as órbitas de um sistema com três ou mais corpos celestes podem exibir comportamento caótico. Apesar de serem determinísticos (e, portanto, previsíveis), esses sistemas são extremamente sensíveis às condições iniciais [5]. Isso causa um problema para a predição das séries temporais originadas por eles, pois uma pequena incerteza na medida afetará toda a previsão.

Tendo em vista o desempenho de modelos não-lineares para previsão de diversas séries temporais [3], optamos por estudar a aplicabilidade de redes neurais artificiais à previsão de séries relacionadas a sistemas com dinâmica caótica. Esta pesquisa comparou o desempenho de quatro arquiteturas de redes neurais artificiais: a rede *Multilayer Perceptron* [2], a rede *Long Short-Term Memory* [3], a rede *Gated Recurrent Unit* (GRU) [6] e, por fim, a rede *Echo State Network* [4].

A comparação foi realizada em quatro cenários de sistemas caóticos, sendo dois destes a tempo discreto e dois a tempo contínuo. No caso, os sistemas a tempo discreto foram o mapa de Hénon [7], e o mapa logístico [8]. Já os sistemas a tempo contínuo foram o sistema de Lorenz [9] e as equações de Mackey-Glass [10]. Nos sistemas multidimensionais, como o mapa de Hénon e o sistema de Lorenz, consideramos apenas uma das variáveis de estado.

Iniciamos a análise através de um processo de *grid-search* para determinarmos os parâmetros ótimos para as redes neurais em cada cenário. Em seguida, utilizando os melhores parâmetros, realizou-se um estudo da progressão do erro quadrático médio (EQM) com o número de amostras de entrada do modelo preditor (nesse caso, chamado de K). Por fim, comparamos qual foi a média e o desvio padrão do EQM com o melhor valor de K de cada modelo nos quatro cenários. No caso, o horizonte de predição utilizado foi $L = 3$. Assim, iremos predizer o valor da terceira iteração à frente do valor atual da série temporal.

Através desta pesquisa, constatamos que a ESN se mostrou uma opção bastante adequada para a previsão. Com efeito, a ESN não apenas alcançou o melhor desempenho em todos os cenários, como também é a estrutura mais leve dentre as consideradas, tendo um processo de treinamento bem mais simples e rápido que as demais redes.

2. Metodologia

2.1. Cenários utilizados

Como o intuito do projeto é analisar o desempenho dos modelos preditores citados em sistemas caóticos, os dados necessários para o treinamento foram obtidos através da simulação numérica dos sistemas caóticos. As próximas seções evidenciam o processo utilizado, junto com os parâmetros escolhidos para cada sistema.

2.1.1. Mapa de Hénon

O mapa de Hénon foi um dos sistemas a tempo discreto escolhidos para esta pesquisa. Esse sistema foi proposto por Michel Hénon em 1976 como um modelo simplificado de uma seção de Poincaré do atrator de Lorenz, sendo descrito pelas equações abaixo [7]:

$$x[n+1] = y[n] + 1 - a \cdot (x[n])^2 \tag{1a}$$

$$y[n+1] = b \cdot x[n] \tag{1b}$$

Para esta pesquisa, foram utilizados os valores usuais para os parâmetros a e b. Logo, têm-se que $a = 1.4$ e $b = 0.3$. Além disso, neste e nos outros sistemas, foi gerado um conjunto de 5000 amostras. A figura 1 mostra a série temporal em $\hat{x}$ e o atrator obtido com a simulação para $[x[0]\ y[0]]^T = [0\ 0]^T$.

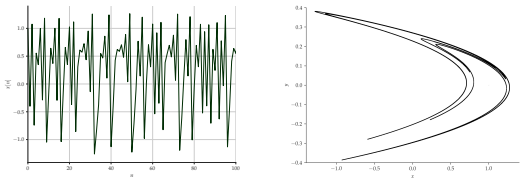


Figura 1: À esquerda, as cem primeiras iterações da série temporal em $\hat{x}$ do mapa de Hénon e, à direita, o atrator correspondente à simulação

2.1.2. Mapa logístico

Conforme foi dito anteriormente, o outro sistema caótico a tempo discreto simulado foi o mapa logístico. Descrito em 1976 por Robert May, o mapa logístico é uma das formas de modelar a população de uma determinada espécie em certos instantes de tempo [8]. A equação a diferenças que descreve esse sistema pode ser vista abaixo:

$$x[n+1] = r \cdot x[n] \cdot (1 - x[n]) \tag{2}$$

Nesse caso, o sistema não chega a operar em caos para qualquer valor de r. Assim, como o estudo visa analisar o desempenho para sistemas caóticos, foi utilizado $r = 3.86$, que, conforme será visto no diagrama de bifurcação abaixo, faz com que a série temporal dada pela equação (2) opere em caos. Novamente, foram simuladas 5000 iterações, para $x[0] = 0.5$ e o resultado pode ser visto na figura 2.

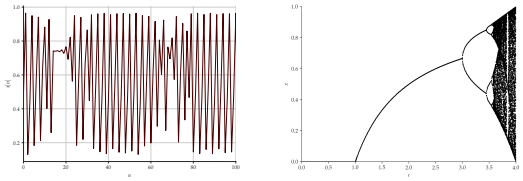


Figura 2: À esquerda, as cem primeiras iterações da série temporal do mapa logístico e, à direita, o diagrama de bifurcação deste sistema

2.1.3. Sistema de Lorenz

O sistema de Lorenz foi um dos sistemas dinâmicos caóticos a tempo contínuo simulados nessa pesquisa. Este sistema foi um dos primeiros grandes trabalhos de sistemas caóticos, sendo considerado por muitos a pesquisa que inaugurou a área [11]. Lorenz modela, através de três equações diferenciais, o fluxo de um fluido em um volume uniformemente aquecido na camada inferior, e uniformemente resfriado na camada superior [9]:

$$\frac{dx}{dt} = -\sigma \cdot (x - y) \tag{3a}$$

$$\frac{dy}{dt} = x \cdot (\rho - z) - y \tag{3b}$$

$$\frac{dz}{dt} = x \cdot y - \beta \cdot z \tag{3c}$$

Para a simulação numérica foi considerado que $\sigma = 10$, $\beta = \frac{8}{3}$, $\rho = 28$ e foi utilizado $dt = 0.01$, gerando 5000 amostras, assim como nos casos discretos. A figura 3 mostra a série temporal em $\hat{x}$ e o atrator de Lorenz para a condição inicial $[x(0)\ y(0)\ z(0)]^T = [0.1\ 0\ 0]^T$.

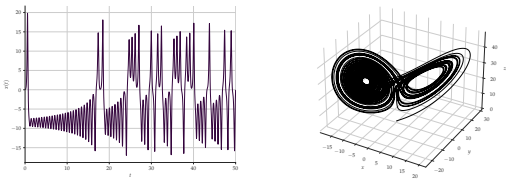


Figura 3: À esquerda, a série temporal em $\hat{x}$ do sistema de Lorenz simulado e, à direita, o diagrama de fases correspondente à simulação

2.1.4. Equações de Mackey-Glass

Por fim, o último sistema caótico simulado, também contínuo, está associado às equações de Mackey-Glass, as quais modelam o controle hormonal da produção de células brancas do sangue e podem ser vistas abaixo [10]:

$$\frac{dP(t)}{dt} = \frac{\beta_0 \cdot \theta^n}{\theta^n + P(t-\tau)^n} - \gamma \cdot P(t) \tag{4a}$$

$$\frac{dP(t)}{dt} = \frac{\beta_0 \cdot \theta^n \cdot P(t-\tau)}{\theta^n + P(t-\tau)^n} - \gamma \cdot P(t) \tag{4b}$$

Neste caso, a equação (4b) exibe comportamento caótico para valores mais altos de τ. Para a simulação numérica, foi utilizado $n = 10$, $\gamma = 0.1$, $\beta = 0.2$, $\theta = 1$, $\tau = 22$, $dt = 1.0$ e $P(0^-) = 0.1$, gerando novamente 5000 amostras. A série e o atrator obtidos podem ser vistos na figura 4.

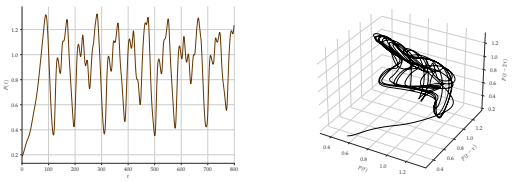


Figura 4: À esquerda, a série temporal da equação (4b) exibida de $t = 0$ a $t = 800$ e, à direita, o atrator correspondente à simulação

2.2. Modelos avaliados e configurações utilizadas

Como foi mencionado na seção introdutória, foi realizado um procedimento de *gridsearch* para a determinação dos melhores parâmetros para as redes neurais. Nesse caso, cada arquitetura testou parâmetros diferentes nesse processo, logo, as próximas seções indicam o que foi testado para cada modelo, além de uma breve exposição sobre as particularidades de cada modelo. Vale reforçar que o *gridsearch* foi realizado em todos os cenários de sistemas caóticos analisados, de forma a obter os melhores parâmetros para cada um deles.

2.2.1. MLP

As redes MLP, como o nome já diz, são compostas por múltiplas camadas de neurônios artificiais *Perceptron* [2]. Cada neurônio recebe x_i atributos de entrada e os pondera por pesos w_i. Em seguida, é aplicada uma função de ativação $\varphi(\cdot)$ sobre esta soma ponderada, acrescida de um termo de *bias*, conforme indica a equação (5).

$$y = \varphi\Big(\sum_{i=1}^{m} w_i x_i + w_0\Big) = \varphi\Big(\sum_{i=0}^{m} w_i x_i\Big) = \varphi(\mathbf{w}^T \cdot \mathbf{x}) \tag{5}$$

Tipicamente, uma rede neural MLP é composta por um número arbitrário N_L de camadas com n neurônios do tipo *Perceptron*, com a característica de que as saídas dos neurônios da l-ésima camada são propagadas para a frente, servindo como as entradas de todos os neurônios da camada seguinte $(l+1)$. Por isso, este tipo de rede é conhecida como totalmente conectada (ou densa) e pertence à categoria de modelos *feedforward*. A figura 5 apresenta a estrutura típica das redes MLP.

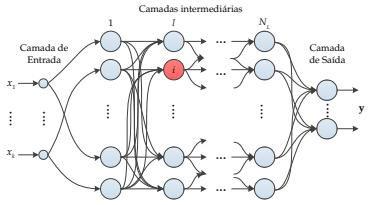


Figura 5: Estrutura típica de uma rede MLP (figura adaptada de [12])

O processo de treinamento de uma rede neural artificial normalmente é realizado com sequências de vetores de entrada $\mathbf{x}$, chamadas de *mini-batch*, sendo que um período de treinamento é chamado de época [13]. Os pesos sinápticos $\mathbf{w}$ são ajustados com um processo iterativo de forma a minimizar uma função custo $J(\mathbf{w})$ que representa uma medida do erro entre as saídas geradas pela rede e as saídas desejadas. Para isso, é frequente o uso de algoritmos de otimização baseados em derivadas da função custo. Para esta pesquisa, foi utilizado o algoritmo Nadam [14]. O processo de retropropagação do erro é empregado para viabilizar o cálculo das derivadas com relação aos pesos sinápticos dos neurônios situados nas camadas internas da rede.

Para as redes *Multilayer Perceptron*, foi testado o tamanho do *batch* para o treinamento, se será ou não utilizado uma camada de normalização do *batch*, a função de ativação para os neurônios *Perceptron* na camada intermediária, a inicialização dos pesos dos neurônios, o número de neurônios na camada intermediária e, por fim, a taxa de aprendizagem.

O *gridsearch* utilizou validação cruzada com 4 *folds* nos dados de treinamento, que correspondiam aos primeiros 4250 valores. Essa proporção entre os dados de treinamento e de teste foi utilizada para todas as redes analisadas.

Vale reforçar que foi utilizada apenas uma camada intermediária em cada cenário. As melhores configurações obtidas podem serem vistas na tabela 1.

Cenário	*Batch normalization*	*Batch size*	Função de ativação	Inicialização	Nº de neurônios	Taxa de aprendizagem
Mapa de Hénon	Não	8	Sigmoid	Glorot Normal	50	0.003
Mapa logístico	Não	2	tanh	Glorot Uniforme	10	0.003
Sistema de Lorenz	Não	2	SELU	Lecun Normal	50	0.001
Equações de Mackey-Glass	Não	4	tanh	Glorot Normal	5	0.001

Tabela 1: Melhores parâmetros para a rede MLP nos cenários em análise

2.2.2. LSTM e GRU

Diferentemente das redes MLP, as redes recorrentes têm estruturas computacionais que podem armazenar os estados anteriores dos neurônios, possuindo também portas não-lineares que regulam o fluxo de informação de entrada e de saída da célula computacional [15].

A diferença entre uma célula LSTM e uma célula GRU é a presença de um vetor de longo prazo $\mathbf{c}$ e um vetor de curto prazo $\mathbf{h}$ na LSTM, conforme indicado na figura 6.

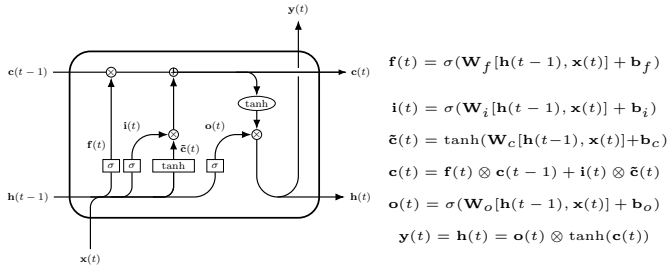


Figura 6: Estrutura e equações de uma célula LSTM

As LSTMs manipulam o vetor $\mathbf{c}(t)$, aprendendo durante o treinamento o que deve ser guardado nele, o que deve ser descartado e o que deve ser aproveitado para gerar a saída $\mathbf{h}(t)$. Dessa forma, podemos dizer que a atualização do vetor de estados $\mathbf{c}(t)$ é feita com o descarte de informações e a incorporação de novidades vindas da entrada.

A célula GRU é uma versão simplificada da célula LSTM e funciona de uma forma bem similar. Nela há somente o vetor de estados de curto prazo $\mathbf{h}$, assim, para que uma memória seja mantida, o conteúdo do local em questão deve ser primeiramente apagado.

À semelhança das redes MLP, o treinamento das redes LSTM e GRU também é realizado através de algoritmos de otimização baseados em derivadas da função custo; a diferença é que agora é necessário propagar as derivadas ao longo da estrutura e, também, ao longo do tempo devido às realimentações. O algoritmo BPTT (*backpropagation-through-time*) representa a extensão do *backpropagation* para o cenário das redes recorrentes [13].

O *gridsearch* para a LSTM e GRU foi bem similar, considerando que essas arquiteturas de redes neurais são semelhantes. A principal diferença com relação ao *gridsearch* da MLP foi o fato de que, ao invés de utilizar-se validação cruzada com $k-folds$, foi utilizado *holdout*. Esse processo dividiu o conjunto de treinamento (novamente composto por 85% dos dados gerados) em 4 seções. Cada seção era composta por uma fração dos dados de treinamento, sendo que cada seção incluía a seção anterior no seu conjunto de dados.

Por exemplo, a segunda seção obtida pelo *holdout* inclui a primeira seção e mais alguns dados, além de uma subdivisão de validação que será utilizada para avaliar o resultado. Com isso, obtém-se conjuntos sequenciais de dados para avaliação do desempenho. Esse procedimento é necessário para as redes recorrentes pois a relação temporal entre os dados de entrada deve ser levada em conta.

Foi avaliado o *batch size*, a inicialização dos pesos, o número de neurônios recorrentes na camada intermediária, e a taxa de aprendizagem, novamente utilizando apenas uma camada intermediária e mantendo a função de ativação usual da célula recorrente ($\tanh$). Por conta disso, para o sistema de Lorenz, foi feito um ajuste de escala para evitar a saturação dos

neurônios.

Os resultados obtidos para a LSTM e para a GRU podem serem vistos nas tabelas 2 e 3, respectivamente.

Cenário	*Batch size*	Inicialização	Nº de neurônios	Taxa de aprendizagem
Mapa de Hénon	4	Glorot Normal	15	0.005
Mapa logístico	2	Glorot Uniforme	100	0.008
Sistema de Lorenz	4	Glorot Uniforme	15	0.003
Equações de Mackey-Glass	2	Glorot Uniforme	50	0.003

Tabela 2: Melhores parâmetros para a rede LSTM nos cenários em análise

Cenário	*Batch size*	Inicialização	Nº de neurônios	Taxa de aprendizagem
Mapa de Hénon	4	Glorot Normal	30	0.003
Mapa logístico	2	Glorot Normal	100	0.003
Sistema de Lorenz	8	Glorot Uniforme	30	0.001
Equações de Mackey-Glass	2	Glorot Uniforme	10	0.005

Tabela 3: Melhores parâmetros para a rede GRU nos cenários em análise

2.2.3. ESN

A *Echo State Network* foi a rede neural mais distinta das outras analisadas nesta pesquisa. Essa rede recorrente baseia-se na teoria de computação por reservatório, chamada assim pois ela utiliza um reservatório de comportamentos dinâmicos que, ao serem combinados linearmente nos neurônios da camada de leitura, produzem as saídas da rede [12].

As equações abaixo descrevem a ESN, sendo $\mathbf{x}(n)$ o vetor de entrada, $\mathbf{W}_{in} \in \mathbb{R}^{N \times K}$ os pesos da camada de entrada, $\mathbf{u}(t)$ o vetor que representa as ativações dos N neurônios não-lineares totalmente conectados do reservatório, representando os estados da rede, $\mathbf{W} \in \mathbb{R}^{N \times N}$ os pesos das conexões recorrentes do reservatório, $\mathbf{f}(\cdot)$ denota as funções de ativação das unidades internas da rede, $\mathbf{W}_{out} \in \mathbb{R}^{L \times N}$ é a matriz dos pesos da camada de saída e, por fim, $\mathbf{y}(n)$ denota as saídas da rede.

$$\mathbf{u}(n+1) = \mathbf{f}(\mathbf{W}_{in} \cdot \mathbf{x}(n+1) + \mathbf{W} \cdot \mathbf{u}(n)) \tag{6a}$$

$$\mathbf{y}(n+1) = \mathbf{W}_{out} \cdot \mathbf{u}(n+1) \tag{6b}$$

O grande diferencial prático desta rede é que os pesos dos neurônios do reservatório $\mathbf{W}$ são ajustados com valores fixos antes do treinamento da camada de saída. Esses parâmetros são determinados tendo em vista a propriedade de estados de eco, existindo alguns métodos simples para a criação aleatória dos pesos e que asseguram a propriedade, conforme demonstrado em [4].

Devido a isso, o ajuste dos neurônios na camada de leitura pode ser realizado com um procedimento de regressão linear, sem necessitar de um treinamento iterativo, como nas redes anteriores. Esse fato economiza tempo de processamento e garante que será atingido o mínimo global da função custo utilizada.

Para esta rede foi realizado um *gridsearch* para determinar o número de neurônios na camada de leitura e o valor para o raio espectral, que determina o maior valor singular da matriz de pesos do reservatório. Por tratar-se de uma rede recorrente, também foi utilizado o processo de *holdout* descrito na seção anterior. Vale também mencionar que a taxa de vazamento, que representa a velocidade com a qual o reservatório atualiza suas dinâmicas, foi fixada em 0.9 para todos os cenários.

A tabela 4 exibe as melhores configurações para a ESN em cada um dos cenários.

Parâmetro	Mapa de Hénon	Mapa logístico	Sistema de Lorenz	Equações de Mackey-Glass
Raio espectral	0.1	0.1	0.2	0.4
Nº de neurônios	500	500	120	500

Tabela 4: Melhores parâmetros para a rede ESN nos cenários em análise

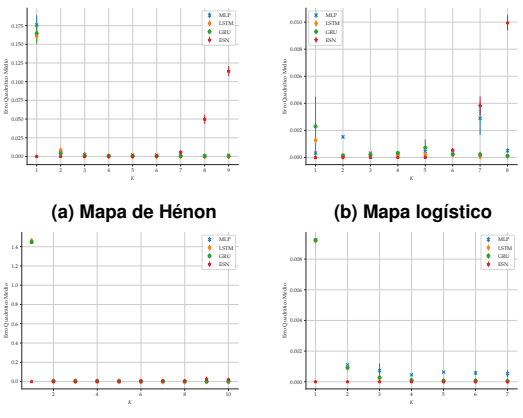
2.3. Análise do melhor valor para K

Com as melhores configurações para cada rede e cenário obtidas, foi analisada a progressão do erro quadrático médio para cada valor de K ~~para cada rede neural, em cada cenário~~. A faixa de valores para K a ser testada foi determinada utilizando a autocorrelação parcial das séries temporais de cada sistema analisado e observando os valores de K que têm as autocorrelações parciais mais relevantes.

Para realizar esse procedimento, cada rede (com as configurações ótimas) foi treinada utilizando 85% dos dados gerados, sendo que 10% dos dados de treinamento foi utilizado como o conjunto de validação (nas redes MLP, LSTM e GRU). Em seguida, com o modelo treinado, foi avaliado o EQM no conjunto de teste (que corresponde aos últimos 750 dados). Esse processo foi realizado 5 vezes para cada K, obtendo assim um valor médio e desvio-padrão para cada K em cada cenário, para cada modelo.

Vale reforçar que, como o ajuste de parâmetros da ESN possui solução em forma fechada, não foi necessário utilizar um conjunto de validação no treinamento. No caso da MLP, LSTM e GRU, o processo iterativo de ajuste dos pesos utilizou um conjunto de validação de forma a seguir o procedimento de *early stopping* para evitar o sobreajuste da rede.

A figura 7 mostra a progressão do EQM para cada K obtida em todos os cenários para todas as redes.



(a) Mapa de Hénon (b) Mapa logístico

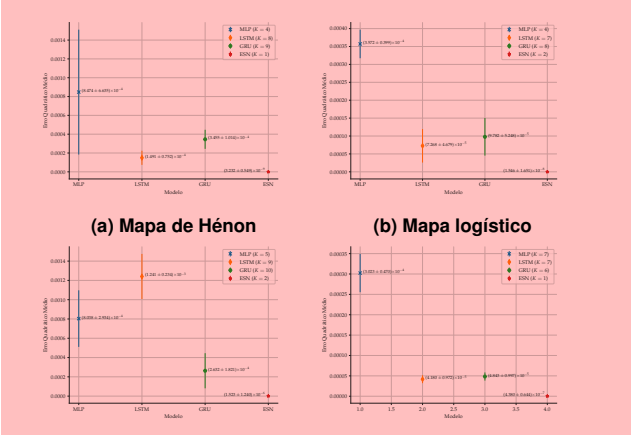
(c) Sistema de Lorenz (d) Equações de Mackey-Glass

Figura 7: Progressão do erro quadrático médio para cada valor de K nos cenários avaliados para todas as redes neurais testadas

3. Resultados

Analisando qual foi o valor de K que produziu o melhor desempenho para cada modelo, realizou-se novamente o processo mencionado na seção anterior e foi obtido o EQM para as melhores configurações (parâmetros e K) para cada modelo, em todos os cenários.

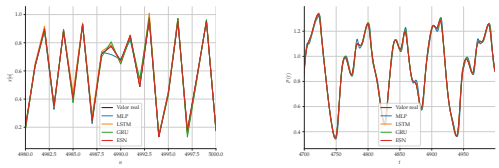
A figura 8 exibe o comparativo dos melhores desempenhos das redes neurais analisadas, e a figura 9 mostra uma comparação da predição em si de cada modelo nos cenários onde a diferença foi mais perceptível visualmente.



(a) Mapa de Hénon (b) Mapa logístico

(c) Sistema de Lorenz (d) Equações de Mackey-Glass

Figura 8: Comparação do melhor desempenho obtido por cada modelo nos cenários testados



(a) Mapa logístico (b) Equações de Mackey-Glass

Figura 9: Comparação da predição realizada por cada modelo nos cenários do mapa logístico e das equações de Mackey-Glass

4. Análise e Conclusão

Analisando os resultados obtidos, percebe-se que, com exceção do cenário do sistema de Lorenz, a rede MLP foi consideravelmente pior do que as redes recorrentes. Também percebe-se que, dentre as redes recorrentes, a ESN obteve um EQM bem inferior ao obtido pela LSTM e pela GRU.

O desempenho inferior da rede MLP com relação às redes recorrentes provavelmente deve-se ao fato de que a relação temporal presente na LSTM, GRU e ESN auxilia na modelagem da dinâmica da série temporal. Já o pior desempenho da LSTM na série temporal do sistema de Lorenz provavelmente está relacionado aos efeitos estocásticos presentes no ajuste de peso dessa rede neural que, conforme indicado em [16], é uma dificuldade do treinamento dessa rede.

Algo interessante de mencionar é que, no geral, o desempenho de todas os modelos estudados foram consideravelmente bons nas séries temporais dos sistemas a tempo contínuo (sistema de Lorenz e equações de Mackey-Glass), sendo que as diferenças foram mais pronunciadas na série temporal do mapa de Hénon e do mapa logístico. Provavelmente, a suavidade presente nas séries de Lorenz e de Mackey-Glass facilita a modelagem do preditor, além do fato de que as séries temporais mencionadas são bem menos erráticas se comparadas às séries de Hénon e do mapa logístico.

O principal resultado obtido foi o fato de que, percebe-se que o modelo preditor utilizando a rede de estado de eco obteve um desempenho bem superior aos outros modelos, em todos os cenários. O erro quadrático médio foi tão baixo que, observando a predição nos dados de teste, praticamente não há diferença entre os valores reais e os valores previstos. Essa superioridade da ESN também é realçada na imagem comparativa do EQM, que indica que a ESN obteve previsões de cerca de 100 a 10000 melhores que os outros modelos.

Esse resultado, aliado ao fato de que o treinamento da ESN é bem menos custoso computacionalmente se comparado ao das outras redes, mostra que esta rede é uma boa alternativa para futuros estudos de modelos preditores de séries temporais. Além disso, conforme mostrado em outros trabalhos como [17, 4, 12], a ESN também é uma boa ferramenta para outras tarefas de extração de informação, como em equalização de canais e separação de fontes.

Como sugestão de trabalhos futuros, pode ser avaliada a eficácia da ESN em reconstruir atratores através de séries temporais experimentais de sistemas caóticos. Se o desempenho para essa tarefa for tão bom quanto o obtido na predição das séries estudadas, a ESN pode-se tornar uma ferramenta ainda mais poderosa para a modelagem de sistemas com dinâmica caótica.

Referências

[1] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[2] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[3] J. T. Connor, R. D. Martin, and L. E. Atlas, "Recurrent neural networks and robust time series prediction," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 240–254, 1994.

[4] H. Jaeger, "Echo state network," *scholarpedia*, vol. 2, no. 9, p. 2330, 2007.

[5] N. Fiedler-Ferrara and C. P. C. do Prado, *Caos: uma introdução*. Editora Blucher, 1994.

[6] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[7] M. Hénon, "A two-dimensional mapping with a strange attractor," *Communications in Mathematical Physics*, vol. 50, pp. 69–77, feb 1976.

[8] R. M. May, "Simple mathematical models with very complicated dynamics," *Nature*, vol. 261, pp. 459–467, jun 1976.

[9] E. N. Lorenz, "Deterministic nonperiodic flow," *Journal of atmospheric sciences*, vol. 20, no. 2, pp. 130–141, 1963.

[10] M. C. Mackey and L. Glass, "Oscillation and chaos in physiological control systems," *Science*, vol. 197, no. 4300, pp. 287–289, 1977.

[11] J. Gleick, *Chaos: The amazing science of the unpredictable*. Vintage Publishing, 1998.

[12] L. Boccato, *Novas propostas e aplicações de redes neurais com estados de eco*. Tese (doutorado), Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação, Campinas, SP, 2013.

[13] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, 2019.

[14] T. Dozat, "Incorporating nesterov momentum into adam," 2016.

[15] S. Haykin, *Neural networks and learning machines, 3/E*. Pearson Education India, 2010.

[16] K. Doya *et al.*, "Bifurcations in the learning of recurrent neural networks 3," *learning (RTRL)*, vol. 3, p. 17, 1992.

[17] H. Jaeger and H. Haas, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication," *science*, vol. 304, no. 5667, pp. 78–80, 2004.